Large scale validation of a computer aided polyp detection algorithm for CT colonography using cluster computing

I. Bitter, PhD, J. E. Brown, BS, D. Brickman, BS, R. M. Summers, MD, PhD

Diagnostic Radiology Department, Warren Grant Magnuson Clinical Center

ABSTRACT

The presented method significantly reduces the time necessary to validate a computed tomographic colonography (CTC) computer aided detection (CAD) algorithm of colonic polyps applied to a large patient database. As the algorithm is being developed on Windows PCs and our target, a Beowulf cluster, is running on Linux PCs, we made the application dual platform compatible using a single source code tree. To maintain, share, and deploy source code, we used CVS (concurrent versions system) software. We built the libraries from their sources for each operating system. Next, we made the CTC CAD algorithm dual-platform compatible and validate that both Windows and Linux produced the same results. Eliminating system dependencies was mostly achieved using the Qt programming library, which encapsulates most of the system dependent functionality in order to present the same interface on either platform. Finally, we wrote scripts to execute the CTC CAD algorithm in parallel. Running hundreds of simultaneous copies of the CTC CAD algorithm on a Beowulf cluster computing network enables execution in less than four hours on our entire collection of over 2400 CT scans, as compared to a month a single PC. As a consequence, our complete patient dataase can be processed daily, boosting research productivity. Large scale validation of a computer aided polyp detection algorithm for CT colonography using cluster computing significantly improves the round trip time of algorithm improvement and revalidation.

Keywords: cluster computing, beowulf, multi platform software development, computed tomographic virtual colonoscopy, polyp detection, computer aided detection, algorithm validation

1. INTRODUCTION

Colorectal cancer is the second leading cause of cancer deaths in the US. Colon carcinomas most frequently develop from colon polyps. Timely screening and removal of pre-cancerous polyps can prevent up to 90% of all colorectal cancer deaths. CT colonography (CTC) offers a minimally invasive screening procedure that requires no intravenous sedatives or recovery time. It also eliminates the risk of bowel punctures. In CTC, radiologists non-invasively screen CT abdomen images for colon polyps. Recent CTC publications report sensitivity and specificity between 80% and 100%, which is equivalent to the gold standard of optical colonoscopy.[1,2] Computer Aided Detection (CAD) of colon polyps is an emerging technology that may improve CTC interpretation results as well as reduce interpretation time, costs, and inter-reader variability. Figure 1 depicts a polyp found by our algorithm, CTCCAD, on a 2D axial and a 3D surface view. To validate CTCCAD and determine its sensitivity and specificity, it is necessary to apply the algorithm to a large patient database with known polyps. This validation is computationally intensive and time consuming. Our database includes 1200 patients, each with two CT scans. With a single CT scan requiring 20 minutes of processing time, processing all datasets would consume a single PCs resources for one month. A Beowulf compute cluster offers the possibility to run several, if not hundreds of, copies of the same application in parallel. The National Institutes of Health (NIH) has such a cluster named Biowulf, consisting of 905 multiprocessor, Linux-based PCs. The purpose of this work is to significantly reduce the time necessary to validate using a large patient database.

ibitter@nih.gov (corresponding author)
jebrown@cc.nih.gov
dbrickman@cc.nih.gov
rms@nih.gov

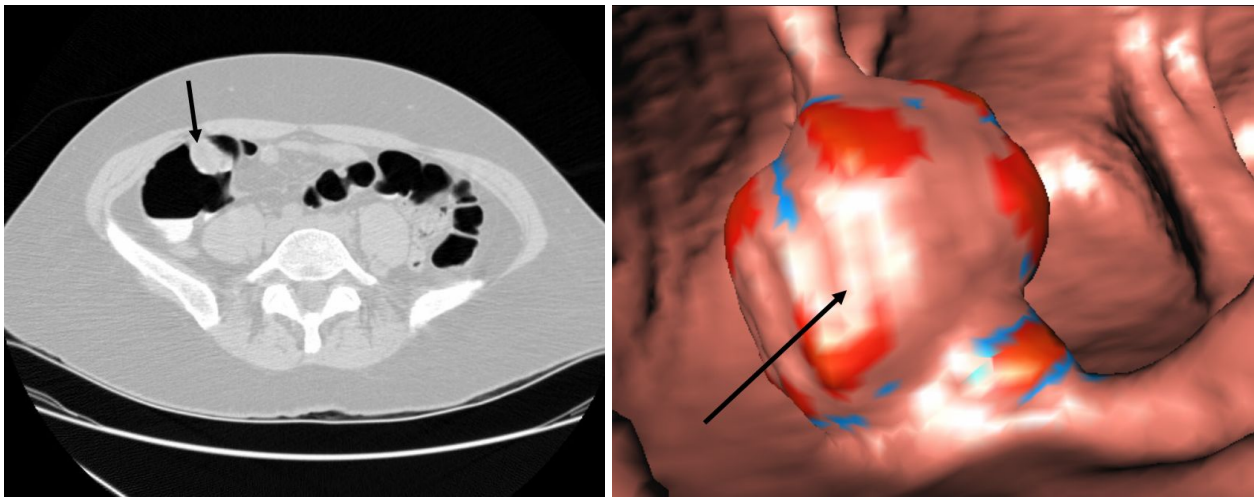

Figure 1. A polyp found by CTCCAD. 2D axial view on the left, 3D surface view on the right.

2. METHODS

2.1. Overview

The first step in preparing for large scale validation was to choose the Beowulf cluster as our validation strategy. As a consequence, all dependant libraries needed to be built in Linux. Next, we made software improvements to insure consistency between Windows and Linux. Finally we created facilities for single and automated batch execution.

2.2. Selection of Validation Strategy

2.2.1. Original Serial Method

Our original infrastructure consisted of several stand-alone, Windows-based development PCs. In order to test the newest version of the algorithm and generate sensitivity and specificity results, one of the PCs had to be designated as a test machine. CTCCAD required up to twenty minutes per CT scan, large scale validation was time prohibited. Hence, we never analyzed more than 150 cases consecutively. As each CT scan occupies approximately 250 MB storage, 2400 scans require approximately 600 GB. These space and time constraints constitutes the major bottleneck for the validation of our algorithm.

2.2.2. Selection of Parallel Method

Parallel execution of our algorithm was an obvious solution to the serial validation bottleneck. We rejected manual distribution for parallel execution of verification tasks, becasue it was cumbersome and could lead to inconsistency errors. We also rejected automated distributed computing over our Windows machines because it relied on idle time from our groups PCs which were heavily in use. A Linux Beowulf computing cluster was available at the NIH. Hence, we concluded that by building a Linux version of CTCCAD, we could easily deploy it to the supercomputing network for rapid execution and verification. The objective was to produce a dual platform version of code that could be built from a single source.

2.3. Implementation

2.3.1. Software Maintenance Tools

We employed CVS, the concurrent versions system, a software tool that maintains the files that change during software development, including a facility for accessing older versions.[3] In addition, it allows multiple users to modify files in parallel, track changes, and merge the results. We changed the locally mounted repository directory setup of CVS to a client server setup. This allows non-Windows computers to access the repository

and automatically handle file format conversions between Windows and Linux. Although CVS is limited to a command line interface, there are graphical interfaces available, such as WinCVS (Windows) and gCVS (Linux) that provide convenient "point-and-click" access to CVS functionality.[4] For C++ compilers, we used Visual Studio for Windows and gcc for Linux.

2.3.2. Software Libraries

The algorithm uses several software libraries, available for both Windows and Linux, to facilitate development. The libraries were built, and their respective install scripts were called from a composite install script.

CTN
The Central Test Node, or CTN, is a reference implementation of the DICOM standard.[5] CTN is used to read and organize the DICOM images. For the Windows build, we used Visual Studio to identify and build only the library project. For the Linux build, we created a script to build the library, but none of the applications.

ITK
ITK provides image processing, registration, and segmentation abilities.[6] We used ITK to smooth the raw DICOM data. ITK was built for both platforms using CMake.[7] In Windows it must be point and click while in Linux it can be scripted.

Qt
Qt provides access to the keyboard and mouse, handles window creation and manipulation, and OpenGL display. Furthermore, it provides excellent string manipulation, XML parsing, file handling and directory services.[8] Building Qt in Windows was done as part of the install program. In Linux, Qt was built through a configure script which separates building and installing of the library, meta-object parser, and demonstration programs. The Linux build needed a few environment variables, all of which are described in the documentation.

Coin3D
Coin3D is a library for 3D graphics representation and rendering, and which complies with the OpenInventor standard.[9] We used it to manipulate and display 3D colon geometry. CoinSoQt merges the functionality of Coin3D and Qt for 3D rendering using OpenGL. All Coin libraries on either platform are built through configure scripts similar to the Qt Linux script.

2.4. Software Improvements

2.4.1. MS-Dependant Code Removal

Creating a single source application required removing Microsoft C++ dependent code or syntax that did not conform to the C++ language standard. As an example, we found several pieces of code which used routines provided in the io.h and direct.h libraries for system and file manipulation. We removed these MS-specific routines and replaced them with Qt functions. This resulted in a single piece of code that was operating system and compiler independent, without the need for any `"ifdef __gcc"` statements.

2.4.2. Standardization of Code and Input Files

Through the use of two compilers and executables, we identified real and potential pitfalls in our algorithm. Corrections to fix syntax and runtime errors in one platform improved the quality of the other. At the end of this process, consistency checks between Windows and Linux builds yielded identical integer computations on both systems, while floating point computations differed by less than 0.01%.

As an example, Windows and Linux use different end of line escape sequences. As a general improvement and solution to the problem, we converted our input parameter files to XML format. XML data is independent of order and does not use white space as a delimiter, and is therefore portable to all platforms. Additionally, XML files have built-in backward compatibility because old programs ignore new tags. They also can be forward compatible if defaults are provided for missing tags.

Figure 2. Biowulf hardware. PCs on the left, blade nodes and network switches on the right.

2.5. Beowulf Cluster

A Beowulf cluster combines multiple computers that may have a mixture of memory, processor capability, on-board cache, and other features.[10] It is typically run over a private internal network, with a single interface to an outside network. Programs can be customized for the Beowulf cluster using the MPI programming library to distribute tasks amongst processors. Alternatively, a single-threaded program can be executed in parallel with different data inputs through a batch script.

The NIH Beowulf cluster is named Biowulf.[11] It consists of over 1800 Intel and AMD processors (905 nodes). Compute nodes vary in memory capacity, ranging from 256MB to 4GB. All nodes are interconnected via 100 Mbit, 1Gbit, or Myrinet networks. User data is stored on RAID file servers, while system files are stored on individual compute nodes. Data IO is slowed by network access to the central file server, while scaling of computational power is limited only by availability of compute nodes.

2.5.1. Dynamic OpenGL Library

Biowulf's compute nodes are intended for computation, not graphics. Hence, OpenGL is not available, but still required by our executable. Since we did not have permission to install the dynamic library on the individual nodes, we place a copy of the OpenGL library in our user directory on the file server. Additionally, we separated all graphical and batch mode code to remove graphical calls from batch execution. After this modification and adding our library directory to the search path, our application successfully executed on each individual node.

2.6. Biowulf Use

2.6.1. Single Batch Execution

We developed a Visual Basic program that generated a script with command line invocations for each data set. On Biowulf, the swarm utility parses the script and submits each command line call as a separate job to the batch system. If more jobs are submitted than nodes available, the batch system dispatches remaining jobs as nodes become free. We sent all jobs to the fast queue, which has a maximum execution time of one hour, but a higher number of available nodes. Options are available to specify processor speed, memory size, maximum execution time, and number of compute nodes. After all batch jobs complete, our data collection script combines the individual output files into a composite file. It also generates a list of all warning and error messages. These files are then sent to a local Windows workstation using the rsync command. The rsync program can be configured to use ssh for automatic authentication and secure transmission.

2.6.2. Automated Batch Execution

We wrote a python script to enable nightly automated verification of the latest version of the algorithm. The script updates the cvs controlled source code and input files on Biowulf and then builds the executable. Next, it assembles a list of valid data directories and creates a matching swarm script. The script waits for the compute nodes to finish execution and searches for directories which failed. Each of the failed directories is re-run once. A report is compiled with a list of directories that finished first, second, and never. The composite program output and the execution report are automatically sent back to a lab PC via rsync. This script is started nightly using cron, a UNIX scheduling program.

3. RESULTS

Running simultaneous copies of the CTCCAD algorithm on the NIH Biowulf cluster computing network enables execution on the entire collection of 2400 CT scans in less than four hours, as compared to the previous execution time of one month with a serial validation strategy.

4. CONCLUSION

CTCCAD is not yet applicable for routine clinical use, but exhibits considerable potential. Large scale validation of a computer aided polyp detection algorithm for CT colonography using cluster computing significantly improves the round trip time of algorithm improvement and re-validation. As a consequence, there is a significant increase in the number of CT scans that can be processed daily. Ultimately, this boosts research productivity and shortens the time to produce publishable results.

ACKNOWLEDGMENTS

We thank the Biowulf administration team for their continued support. We also thank Drs. Perry Pickhardt, Richard Choi and William Schindler for CT colonography data.

REFERENCES

1. P. J. Pickhardt, J. R. Choi, I. Hwang, J. A. Butler, M. L. Puckett, H. A. Hildebrandt, R. K. Wong, P. Nugent, P. A. Mysliwiec, and W. R. Schindler, "Computed tomographic virtual colonoscopy to screen for colorectal neoplasia in asymptomatic adults," *N Engl J Med* **349**, pp. 2191–2200, 2003.
2. H. M. Fenlon, D. P. Nunes, P. C. S. 3rd, M. A. Barish, P. D. Clarke, and J. T. Ferrucci, "A comparison of virtual and conventional colonoscopy for the detection of colorectal polyps [published erratum appears in N Engl J Med 2000," *N Engl J Med* **341**, pp. 1496–1503, 1999.
3. *CVS*, http://www.cvshome.org.
4. *WinCVS*, http://www.wincvs.org.
5. *CTN*, http://wuerlim.wustl.edu/DICOM/ctn.html.
6. *ITK*, http://www.itk.org.
7. *CMake*, http://www.cmake.org.
8. *Qt*, http://www.trolltech.com.
9. *Coin3D*, http://www.coin3d.org.
10. *Beowulf*, http://www.beowulf.org.
11. *Biowulf*, http://biowulf.nih.gov.